The Data Grid

Towards an Architecture for the Distributed Management of Large Scientific Databases

by Chernevak, Foster, Kesselman, Salisbury, and Tuecke

presented by Asad B. Sayeed on this twentieth day of the month of October, anno domini 2005
Large data collections
- Appear in many scientific domains: climate studies, physics, genomics.
- Terabyte, nearly petabyte sizes.
- Users AND resources often very distributed geographically, large community.

Problematic combination: large data, geography, computationally intensive analysis with many queries.

Numerous solutions to individual issues exist, but no overarching architecture.

Hence, *data grid*. 
Specialization and extension of “Grid” paradigm:
- Integrating infrastructure for distributed computation.

Complements other activities
- High speed disk caches
- Tertiary storage, cache management
- Metadata standards and retrieval mechanisms: from digital library community.
- Etc

Structure of paper
- Discusses principles in designing data grid architecture
- Basic services: storage systems and metadata management.
- Higher-level services and implementation issues.
Data Grid Design

- **Mechanism Neutrality**
  - DG architecture to be independent of lower-level storage/access systems. *Encapsulation* of local peculiarities.

- **Policy Neutrality**
  - Expose design decisions with performance implications to the user; ie let user set priorities.

- **Compatibility with Grid infrastructure**
  - Exploit Grid services like authentication and resource management.

- **Uniformity of information infrastructure**
  - Use the same data model and interface as required to handle the underlying Grid infrastructure.
Layered architecture based on these four principles.

Lower layers provide high-performance access without imposing any policy choices—promote reuse of basic efficient mechanisms.
Two fundamental services: data access vs. metadata access.

Data Grid: keep data and metadata separate. This maximizes flexibility (applications that want them to be the same can still pretend).

Storage systems
- Heterogeneous systems, locations: mechanism neutral.
- Unit of info: file instances. Can be represented as OS files, DB entries, etc.
- Logical notion of storage systems: can be implemented as HTTP servers and other diverse things.
Data access

- Obvious basic functionality: read/write file instances, determine and set file instance characteristics.
- Support replica management optimizations—allow transfer of data directly between storage systems.
- Data grid complications
  - Heterogeneous security environments.
  - Reservation capabilities on storage systems and environments when increased performance is required.
  - Performance monitoring and self-optimization.
  - Error detection and reporting.
Metadata service

- Metadata: information about the data grid itself, ie storage systems, file instance locations, file instance contents, and so on.

- Types of metadata
  - Application metadata: info from eg. Scientific applications
  - Replica metadata: location of data, data replicas, and criteria for selecting storage and access locations.
  - System configuration metadata: information about grid. Eg, connectivity, storage capacity, usage policy.

- Service: single interface.

- Application queries sent to repository/catalog.
  Repository consists of references to logical files, which can be mapped to actual replicas.
Metadata complications

- Difficult to select uniform representation and interface: numerous existing metadata representations reflecting different needs and philosophies (XML, indexing data structures, etc).
- Large scale metadata issues: scalability, heterogeneity, distributed environment, ownership and local control over data access, robustness in the face of partial failures.
- Hence: hierarchical and distributed system.
- Similar systems: LDAP, existing Grid metadata systems.
Other basic services

- Authorization/authentication—already exists in Grid.
- Resource reservation, co-allocation—predictable performance
- Performance measurements, estimation techniques
- Instrumentation services for storage transfers and other operations.
Higher-Level Data Grid Components

- Potentially unlimited number of possible components.
- Two important components: replica management and replica selection.
- Replica management
  - Replica Manager—creates/deletes replicas of file instances.
  - Replicas are a “user-asserted” data correspondence between files: not necessarily byte equality.
  - Use of repository/catalog mentioned previously—map logical files to physical replicas.
  - Replica Manager does not contain storage location criteria—leave policy matters to the application.
Replica Selection and Data Filtering

- Must furnish applications with replica with the best access performance characteristics.
- Criteria: speed, cost, security
- Can trigger creation of better replicas.
- Use Grid information to obtain network performance data, rank replicas.
- Possible feature: extract important data subsets out of large file instances as replicas of their own.
  - This requires special filtering programs that understands how to analyze the large files.
Implementation Experiences

- LDAP implementation
  - Used LDAP to construct catalogs/repositories.
  - Catalog: tree structure known as Directory Information Tree (DIT).
  - Two applications prototyped: Climate modeling, data visualizations.
  - Climate modeling:
    - DIT: root node, node for each collection (four), node for each logical file per collection.
    - Metadata: XML. Data accessed via URLs.
    - Prototype: user manually chooses replica.
LDAP implementation

- Data visualization application
  - Desktop client streams remote data
  - Each file: a timestep in a series of eg astronomical data.
    - Can be of different resolutions.
    - Can be of different data layouts (endianness, etc).
  - Several thousand files.
  - Each file has multiple replicas listed in catalog.
  - Their implementation scales poorly: too many objects in replica catalog, requires distributed catalog implementation.
  - Solution: organize logical files into collections.
    - Associate location info with collection.
    - Single lookup finds path to logical file for collection, which the can be mapped to replica.
    - Greatly reduces storage space.
    - (Apparently not tested...)

Another lesson: keep metadata and replica management separate.

- Replica and Metadata Catalogs should be distinct.
- Data contents info should be stored in Metadata Catalog.
- Data location info/mapping should be stored in Replica Catalog.
Implementation Experiences

Figure 2: The structure of a replica catalog.
“Made progress” on defining required basic services for data grid—preliminary data access API.

Prototype: contains interfaces to local file access, HTTP, FTP, DPSS network disk caches.

Implemented (as above) replica management and metadata services via LDAP.